

AD-A073 974

HARRY DIAMOND LABS ADELPHI MD  
QINTEG--A PROGRAM FOR COMPUTING GAUSSIAN QUADRATURE COEFFICIENT--ETC(U)  
AUG 79 T H HOPP  
HDL-TR-1890

F/G 9/2

UNCLASSIFIED

NL

| OF |

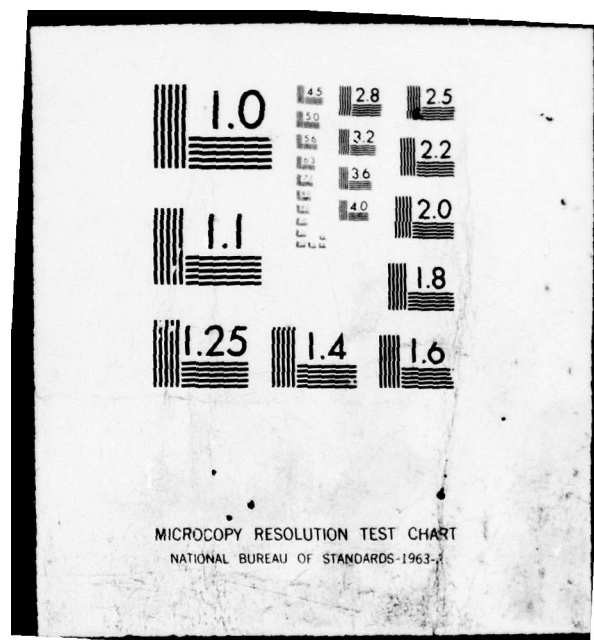
AD  
AO 73974



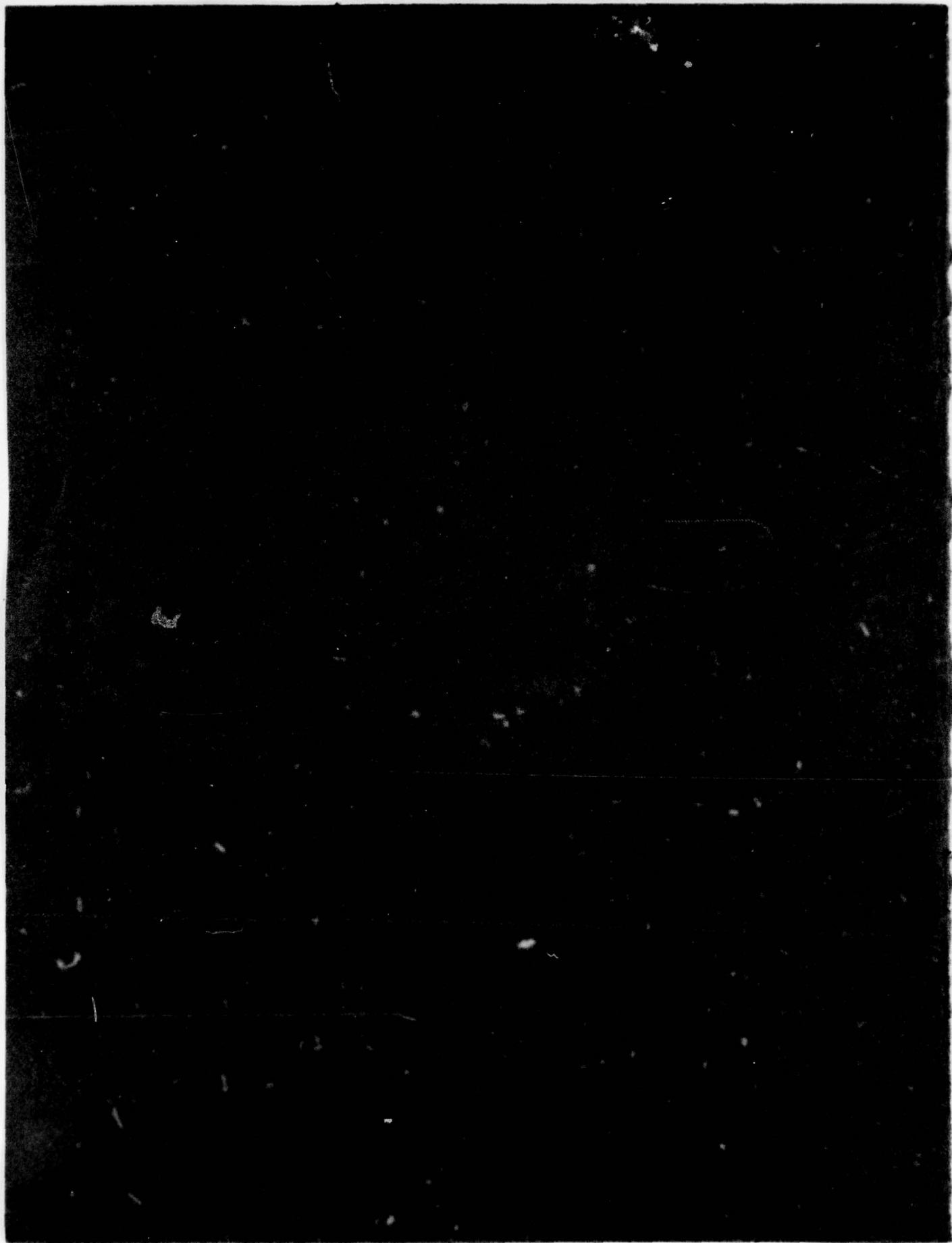
END  
DATE  
FILMED

10-19

DOC



DA073974



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER HDL-TR-1890	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) QINTEG--A Program for Computing Gaussian Quadrature Coefficients.	5. TYPE OF REPORT & PERIOD COVERED Technical Report.	
7. AUTHOR(s) Theodore H. Hopp		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Harry Diamond Laboratories 2800 Powder Mill Road Adelphi, MD 20783		8. CONTRACT OR GRANT NUMBER(s) DA: 1L161102AH44
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Materiel Development and Readiness Command Alexandria, VA 22333		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Ele: 6.11.02.A
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE August 1979
		13. NUMBER OF PAGES 35
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)  Approved for public release, distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES HDL Project: A44813 DRCMS Code: 611102.H44011		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Quadrature Gaussian quadrature Numerical integration Orthogonal polynomials Gaussian integration		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report presents a program (QINTEG) to compute Gaussian quadrature coefficients for integration with respect to a weighting function. Discontinuities, singularities, or oscillatory behavior of integrands make many problems unsuited to numerical solution using existing general purpose integration programs. When the undesirable behavior of the integrand can be absorbed into the weighting function, numerical integration using the coefficients computed by QINTEG can be accomplished. Improvement		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

163 050

next  
page

JOD

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

of nearly two orders of magnitude in execution time over an existing program was achieved for a problem involving oscillatory integrands. A program listing is included.

Accession For		<input checked="checked" type="checkbox"/>
NTIS GRA&I		<input type="checkbox"/>
DDC TAB		<input type="checkbox"/>
Unannounced		<input type="checkbox"/>
Justification		<input type="checkbox"/>
By _____		
Distribution/		
Availability Codes		
Dist	Avail and/or	special
<b>A</b>		

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

## CONTENTS

	<u>Page</u>
1. INTRODUCTION . . . . .	5
2. BACKGROUND THEORY . . . . .	5
3. COMPUTATION OF WEIGHTS AND ABSCISSAE . . . . .	10
4. FOURIER-LIKE INTEGRALS . . . . .	17
5. SUMMARY . . . . .	26
ACKNOWLEDGEMENTS . . . . .	26
LITERATURE CITED . . . . .	27
APPENDIX A.--QINTEG COMPUTER PROGRAM LISTING . . . . .	29
DISTRIBUTION . . . . .	35

## TABLES

I	Abcissae and Weights for Gaussian Integration with Weighting Function $w(x)$ : 10 cycles . . . . .	23
II	Abcissae and Weights for Gaussian Integration with Weighting Function $w(x)$ : 20 cycles . . . . .	23
III	Abcissae and Weights for Gaussian Integration with Weighting Function $w(x)$ : 40 cycles . . . . .	24
IV	Abcissae and Weights for Gaussian Integration with Weighting Function $w(x)$ : 80 cycles . . . . .	24
V	Abcissae and Weights for Gaussian Integration with Weighting Function $w(x)$ : 160 cycles . . . . .	25
VI	Abcissae and Weights for Gaussian Integration with Weighting Function $w(x)$ : 320 cycles . . . . .	25

## 1. INTRODUCTION

This report describes an algorithm and a computer program for general Gaussian numerical integration (quadrature). The algorithm is based on developing approximations of the form

$$\int_a^b w(x) f(x) dx \approx \sum_{i=1}^n H_i f(x_i)$$

In this paper, we view the integration as the integral of  $f(x)$  with respect to a weighting function  $w(x)$ , rather than the integral of a function  $w(x)f(x)$ . These approximations can be developed for a wide class of weighting functions. For a somewhat more restricted class of functions, these approximations significantly improve program execution time over the time needed by more traditional programs.

The program to be described was developed as part of an investigation of the coherence properties of reflected laser light. This investigation involved the evaluation of integrals with highly oscillatory integrands. By appropriately incorporating the oscillatory factors of the integrand in the weighting function, the execution time for evaluating these integrals was reduced by nearly two orders of magnitude from that needed by an existing classical Gaussian integration program.<sup>1</sup>

The program developed in this report has wider applicability than removing oscillatory factors of an integrand. Many numerical integration programs either cannot be used at all or are unreliable when an integrand contains discontinuities or singularities in the interval of integration. The algorithm presented here can be used reliably when the undesirable behavior of the integrand can be incorporated in the weighting function. Conditions under which this can be done are given in section 2.

## 2. BACKGROUND THEORY

By far the most common approach to evaluating integrals of the form

$$I = \int_a^b w(x) f(x) dx$$

consists of approximating  $f(x)$  by some polynomial  $P_{n-1}(x)$  of degree  $n - 1$  and evaluating

---

<sup>1</sup>A. Hausner and J. D. Hutchinson, FOGIE: An Adaptive Code for Numerical Integrals Using Gaussian Quadrature, Proc. 1975 Army Numerical Analysis and Computer Conference, Army Research Office Report 75-3 (1975), 139-177.

$$I \approx \int_a^b w(x) P_{n-1}(x) dx .$$

The polynomial  $P_{n-1}(x)$  is obtained by choosing  $n$  points  $\{x_i | i = 1, \dots, n\}$  and forcing  $P_{n-1}(x_i) = f(x_i)$  for all  $i$ .  $P_{n-1}(x)$  is termed a collocation polynomial for  $f(x)$ . Typical of integration schemes is to require  $\{x_i\}$  to be equally spaced over the interval  $[a, b]$  (such as Simpson's rule) or require that  $\{x_i\}$  be chosen so that  $I$  can be approximated as an unweighted average of  $\{P_{n-1}(x_i)\} (= \{f(x_i)\})$  (such as Chebyshev's rule). This section outlines the theory of orthogonal polynomials as it applies to numerical integration and approximation. It then shows how this theory leads to an "optimal" choice for abscissae  $\{x_i\}$  and weights  $\{H_i\}$ .

We consider a sequence of polynomials  $\{Q_0(x), Q_1(x), \dots\}$ , where  $Q_n(x)$  is a polynomial of degree  $n$  in  $x$  with real coefficients. We define such a system to be orthogonal on  $(a, b)$  with respect to the weighting function  $w(x)$  if

$$\int_a^b w(x) Q_m(x) Q_n(x) dx = \begin{cases} 0 & , m \neq n, m, n \geq 0 \\ h_n^2 & , 0 < h_n^2 < \infty, m = n \geq 0 . \end{cases}$$

For a given interval  $(a, b)$ , such a system exists if and only if two conditions hold for the weighting function. The first is that

$$w(x) \geq 0, \quad a \leq x \leq b ,$$

and the second condition is that all integrals of the form

$$\int_a^b x^i w(x) dx$$

must exist.

Several properties of systems of orthogonal polynomials allow development of integration schemes. First, an arbitrary polynomial of degree  $n$  can be expressed as a linear combination of  $Q_i(x)$ ,  $i = 0, 1, \dots, n$ . As a result of this, any orthogonal polynomial  $Q_n(x)$  is orthogonal (with respect to the weighting function) to all polynomials of degree less than  $n$ . Another property of orthogonal polynomials is that each  $Q_n(x)$  has  $n$  distinct real roots, all of which lie in the orthogonality interval. This property is very important since the roots of  $Q_n(x)$  become  $\{x_i\}$  for approximating integrals with an  $n$ -point formula. Finally, a system of

orthogonal polynomials is uniquely specified to within a multiplicative constant by the weighting function and the interval of integration.

From these properties of orthogonal polynomials, an interesting result can be derived. Firstly, evaluating an integral  $I$  by an  $n$ -point summation is equivalent to approximating  $f(x)$  by a collocation polynomial of degree at least  $n - 1$ . Abscissae  $\{x_i\}$ ,  $i = 1, \dots, n$ , can be taken as the roots of  $Q_n(x)$ , the  $n$ th degree orthogonal polynomial.  $P_A(x)$  and  $P_B(x)$  are two collocation polynomials for  $f(x)$  of order less than  $2n$  such that

$$P_A(x_i) = P_B(x_i) = f(x_i), \quad i = 1, \dots, n,$$

(from the definition of collocation polynomials). Because of this, we observe that

$$P_A(x) - P_B(x) = Q_n(x) \pi_{n-1}(x),$$

where  $\pi_{n-1}(x)$  is a polynomial of order at most  $n - 1$ . ( $Q_n(x)$  is a factor of the difference polynomial because each  $x_i$  is a root of  $P_A(x) - P_B(x)$ . Also, since  $P_A(x)$  and  $P_B(x)$  are assumed to be polynomials of order less than  $2n$  and  $Q_n(x)$  is of order  $n$ ,  $\pi_{n-1}(x)$  can be of order at most  $n - 1$ .) Considering the difference in the estimates of the integral using collocation polynomials  $P_A(x)$  and  $P_B(x)$ ,

$$\int_a^b w(x) P_A(x) dx - \int_a^b w(x) P_B(x) dx = \int_a^b w(x) Q_n(x) \pi_{n-1}(x) dx = 0.$$

The last step follows because of the orthogonality property of the orthogonal polynomials. This is the desired result. It shows that if  $\{x_i\}$  are the roots of orthogonal polynomial  $Q_n(x)$ , all collocation polynomials of order  $2n - 1$  or less collocated with  $f(x)$  at  $\{x_i\}$  yield the same estimate for the integral.

Now a formula may be developed for the approximation to the integral. Once the order,  $n$ , of the approximation is fixed, the Lagrange interpolation formula<sup>2</sup> can be used as a collocation polynomial of degree  $n - 1$ ,

$$L(x) = \sum_{i=1}^n \ell_i f(x_i),$$

---

<sup>2</sup>Petr Beckmann, *Orthogonal Polynomials for Engineers and Scientists*, The Golem Press, Boulder, CO (1973).

where

$$l_i = \frac{(x - x_1)(x - x_2) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_1)(x_i - x_2) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}.$$

This polynomial can be written<sup>3</sup>

$$L(x) = \sum_{i=1}^n \frac{Q_n(x)}{(x - x_i)Q'_n(x_i)} f(x_i),$$

where  $Q'_n(x_i)$  denotes the first derivative of  $Q_n(x)$  evaluated at  $x_i$ . Our estimate of the original integral can now be expressed as

$$\begin{aligned} \int_a^b w(x)f(x) dx &\approx \int_a^b w(x)L(x) dx = \int_a^b w(x) \sum_{i=1}^n \frac{Q_n(x)}{(x - x_i)Q'_n(x_i)} f(x_i) dx \\ &= \sum_{i=1}^n f(x_i) \left[ \frac{1}{Q'_n(x_i)} \int_a^b \frac{w(x)Q_n(x)}{x - x_i} dx \right], \end{aligned}$$

which can be written

$$\int_a^b w(x)f(x) dx \approx \sum_{i=1}^n H_i f(x_i),$$

where

$$H_i = \frac{1}{Q'_n(x_i)} \int_a^b \frac{w(x)Q_n(x)}{x - x_i} dx.$$

Given a weighting function, an integration interval, and a system of orthogonal polynomials for  $w(x)$  on the interval, the  $\{x_i\}$  and  $\{H_i\}$  can be calculated independently of  $f(x)$ . Additionally, the approximation is exact for all polynomials of order  $2n - 1$  or less. This suggests that the error of the approximation is bounded by

<sup>3</sup>M. Abramowitz and I. A. Stegun, ed., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards Applied Mathematics Series, 55 (1970).

$$C_n \inf[f^{(2n)}(\xi)] \leq \left| \int_a^b w(x) f(x) dx - \sum_{i=1}^n H_i f(x_i) \right| \leq C_n \sup[f^{(2n)}(\xi)], \quad a < \xi < b,$$

for some constant  $C_n$ , a result that can be rigorously derived.<sup>4</sup>

The approximate integration as developed above is optimal in the following sense: no approximation exists for which the error of the approximation is bounded by a higher order derivative of the function being integrated. As intuitive argument to support this statement is that, since we have  $n$  abscissae and  $n$  weights for the approximation,  $f(x)$  is being fit with  $2n$  parameters. The highest order polynomial that can be fit exactly with  $2n$  parameters is of degree  $2n - 1$ . Thus, no integration scheme using  $2n$  parameters can exactly estimate the integral of arbitrary polynomials of degree greater than  $2n - 1$ . The error cannot be bounded by a higher order derivative of  $f(x)$ , for that would imply that the error for polynomials of degree  $2n$  would be bounded on both sides by zero.

The derivation of the integration scheme in terms of collocation polynomials suggests why oscillatory integrands pose a problem for numerical integration. How accurately the integration is estimated depends in general on how accurately the integrand is approximated by the collocation polynomial. If the collocation polynomial is to approximate an oscillatory function, many collocation points must be chosen in the interval of integration to keep the collocation polynomial close to the integrand function. This problem is closely related to the problem of sampling in signal processing applications.<sup>5</sup> Results from sampling theory state that to uniquely define a frequency limited function in an interval of  $N$  cycles of the highest frequency component,  $2N + 1$  sample points are needed. Although only approximating an oscillatory integrand with a collocation polynomial is of interest, the number of collocation points needed to reduce the error of the approximation below a given level is a function of the number of cycles of the integrand in the interval. As the number of cycles of the integrand increases, the number of collocation points also must increase; this increase leads to costly and at times unreliable estimates of the integral.

<sup>4</sup>P. J. Davis and P. Rabinowitz, *Numerical Integration*, Blaisdell, Waltham, MA (1967).

<sup>5</sup>J. W. Tukey, *The Estimation of (Power) Spectra and Related Quantities*, in *On Numerical Approximation*, R. E. Langer, ed., University of Wisconsin Press, Madison, WI (1959), 389-411.

The integration scheme outlined in this section alleviates the problem of oscillatory integrands when the oscillating factor can be absorbed into the weighting function. A way of absorbing oscillatory behavior into a weighting function is detailed in section 4. In section 3, we derive the algorithms necessary to compute abscissae and weights once a weighting function and an interval of integration are established.

### 3. COMPUTATION OF WEIGHTS AND ABCISSAE

This section details specific algorithms to compute integration weights and abscissae from a weighting function and an interval of integration. Approximations of the form

$$\int_a^b w(x) f(x) dx \approx \sum_{i=1}^n H_i f(x_i)$$

are being developed. Order  $n$  of the estimate is chosen beforehand. In section 2, the  $\{x_i\}$  serve as the roots of  $Q_n(x)$ , the  $n$ th orthogonal polynomial on  $(a,b)$  with respect to  $w(x)$ . The weights are then given by

$$H_i = \frac{1}{Q'_n(x_i)} \int_a^b \frac{Q_n(x)}{x - x_i} w(x) dx .$$

There have been several approaches to the problem of calculating  $\{x_i\}$  and  $\{H_i\}$  from the above equations. For instance, Gautschi<sup>6</sup> describes an algorithm that develops a convergent sequence of approximations to the abscissae and the weights. The algorithms developed in this report can be considered a brute force approach. The problem itself is inherently ill-conditioned.<sup>7</sup> However, the availability of extended precision arithmetic on modern computers allows a straightforward solution to yield an answer of sufficient precision to be useful in many applications.

The problem here is to calculate the coefficients of  $Q_n(x)$ , obtain the  $n$  roots of  $Q_n(x)$ , and finally compute  $\{H_i\}$ . Since many library programs exist for calculating roots of polynomials, the second step of the problem is not of concern in this report. The algorithm for computing  $Q_n(x)$  may now be derived.

<sup>6</sup>W. Gautschi, Algorithm 331--Gaussian Quadrature Formulas, CACM, 11, No. 6 (June 1968).

<sup>7</sup>W. Gautschi, Construction of Gauss-Christoffel Quadrature Formulas, Mathematics of Computation, 22 (1968), 251-270.

First,  $Q_n(x)$  can be multiplied by any constant without the roots or  $\{H_i\}$  being changed. Thus, the leading coefficient [the coefficient of  $x^k$  in  $Q_k(x)$ ] of the orthogonal polynomials may be set to unity. Under this normalization, there is a classic recursion relationship for orthogonal polynomials.<sup>2</sup>

We have

$$\begin{cases} Q_0(x) = 1, \\ Q_1(x) = x - B_0, \\ Q_{i+1}(x) = (x - B_i)Q_i(x) - \frac{h_i^2}{h_{i-1}^2} Q_{i-1}(x), \quad i \geq 1, \end{cases}$$

where

$$h_i^2 = \int_a^b w(x) Q_i^2(x) dx$$

is the squared norm of  $Q_i(x)$  and

$$B_i = h_i^{-2} \int_a^b x w(x) Q_i^2(x) dx$$

is the first moment of  $Q_i^2(x)$ . This recursion relationship is the basis for calculating the coefficients of  $Q_n$ .

The notation  $C_{i,j}$  ( $0 \leq j \leq i$ ) denotes the coefficient of  $x^j$  in  $Q_i(x)$ . Thus,

$$Q_i(x) = \sum_{j=0}^i C_{i,j} x^j.$$

( $C_{i,i} = 1$  is the normalizing condition.) We also generalize the definition of  $C_{i,j}$  by setting  $C_{i,j} = 0$  for  $i < 0$ ,  $j < 0$ , or  $j > i$ . When the expansion of  $Q_i(x)$  is substituted in the equation for  $h_i^2$ ,

---

<sup>2</sup>Petr Beckmann, *Orthogonal Polynomials for Engineers and Scientists*, The Golem Press, Boulder, CO (1973).

$$\begin{aligned}
h_i^2 &= \int_a^b w(x) \left( \sum_{j=0}^i c_{i,j} x^j \right) Q_i(x) dx \\
&= \int_a^b w(x) x^i Q_i(x) dx + \int_a^b w(x) \left( \sum_{j=0}^{i-1} c_{i,j} x^j \right) Q_i(x) dx \\
&= \int_a^b w(x) x^i Q_i(x) dx \\
&= \sum_{j=0}^i \left( c_{i,j} \int_a^b w(x) x^{i+j} dx \right) .
\end{aligned}$$

The third step follows because  $Q_i(x)$  is orthogonal to

$$\sum_{j=0}^{i-1} c_{i,j} x^j ,$$

which is a polynomial of degree  $i - 1$ . In the same manner, one can derive

$$B_i = c_{i,i-1} + h_i^{-2} \sum_{j=0}^i \left( c_{i,j} \int_a^b w(x) x^{i+j+1} dx \right)$$

Thus, to evaluate  $h_i^2$  and  $B_i$  for any  $Q_i(x)$ ,

$$E_j = \int_a^b w(x) x^j dx$$

must be available for  $0 \leq j \leq 2i + 1$ . Then

$$h_i^2 = \sum_{j=0}^i c_{i,j} E_{i+j}$$

and

$$B_i = c_{i,i-1} + h_i^{-2} \sum_{j=0}^i c_{i,j} E_{i+j+1} .$$

The quantities  $E_j$  must be easily calculable for all necessary  $j$ . Since eventually  $B_{n-1}$  will be calculated, values of  $E_j$  must be available for  $0 \leq j \leq 2n - 1$ . Then the following algorithm obtains for computing the coefficients of  $Q_n(x)$ :

- a. Calculate  $E_j$ : Evaluate  $E_j$  for  $0 \leq j \leq 2n - 1$ .
- b. Initialize recursion: Set  $C_{0,0} \leftarrow 1$  [ $Q_0(x) = 1$ ]; set  $h_0^2 \leftarrow E_0$ ,  $B_0 \leftarrow E_1/h_0^2$ ; set  $C_{1,1} \leftarrow 1$ ,  $C_{1,0} \leftarrow -B_0$  [ $Q_1(x) = x - B_0$ ]; set the recursion counter  $i \leftarrow 1$ .
- c. Check termination: At this point, the coefficients for  $Q_i(x)$  have been obtained. If  $i = n$ , stop with the answer. Otherwise, continue to step d.

- d. Calculate  $h_i^2$  and  $B_i$ : Set

$$h_i^2 \leftarrow \sum_{j=0}^i C_{i,j} E_{i+j};$$

set

$$B_i \leftarrow C_{i,i-1} + \left( \sum_{j=0}^i C_{i,j} E_{i+j+1} \right) / h_i^2.$$

- e. Perform recursion: Calculate the coefficients  $Q_{i+1}(x)$ . For each  $j$ ,  $0 \leq j \leq i + 1$ , set

$$C_{i+1,j} \leftarrow C_{i,j-1} - B_i C_{i,j} - h_i^2 C_{i,i-1,j} / h_{i-1}^2.$$

- f. Loop back: Set  $i \leftarrow i + 1$  and go to step c.

The above algorithm is easily programmable. Caution must be observed in step e, where the notation uses the generalization of the coefficients to  $C_{\ell,m} = 0$  for  $m > \ell$  when  $j = i$  and  $j = i + 1$ .

The next step is to obtain the roots of  $Q_n(x)$ . Canned programs for performing this function are readily available in most numerical analysis subroutine libraries.<sup>8,9</sup> Thus, calculating the  $\{x_i\}$  is trivial. All the  $x_i$  are real, distinct, and inside the interval  $(a,b)$ .

<sup>8</sup>IMSL Library 1, Editor 6 (FORTRAN IV), IBM S/370-360, Xerox Sigma S/6-7-9-11-560 (1977).

<sup>9</sup>IBM System/360 Scientific Subroutine Package, Version III, Programmers Manual, IBM GH20-0205-4 (August 1970).

Now the  $H_i$  can be calculated for each  $x_i$ . One can easily calculate  $Q'_n(x_i)$ . Since

$$Q_n(x) = \sum_{j=0}^n c_{n,j} x^j,$$

then

$$Q'_n(x) = \sum_{j=0}^{n-1} (j+1) c_{n,j+1} x^j.$$

Thus,

$$Q'_n(x_i) = \sum_{j=0}^{n-1} (j+1) c_{n,j+1} x_i^j,$$

which is easily evaluated.

The remaining step is to evaluate the integral

$$\int_a^b \frac{Q_n(x)}{x - x_i} w(x) dx.$$

This integral is expressed in terms of  $E_j$ . However, first a method must be developed to obtain the coefficients of the deflated polynomial  $Q_n(x)/(x - x_i)$ . The notation  $\bar{c}_{n,j}$  denotes the coefficient of  $x^j$  in the deflated polynomial. (The coefficients are functions of  $x_i$ .) Thus, by definition,

$$\frac{Q_n(x)}{x - x_i} = \sum_{j=0}^{n-1} \bar{c}_{n,j} x^j$$

( $\bar{c}_{n,j} = 0$  for  $j < 0$  or  $j > n - 1$ .) We consider the equation

$$\frac{Q_n(x)}{x - x_i} = \frac{Q_n(x)}{x} \frac{1}{1 - \frac{x_i}{x}}.$$

Ignoring for the moment any questions of convergence, one can expand this expression in a power series in  $x_i/x$ ,

$$\frac{Q_n(x)}{x - x_i} = \frac{Q_n(x)}{x} \left[ 1 + \frac{x_i}{x} + \left(\frac{x_i}{x}\right)^2 + \dots \right],$$

and rewrite it as

$$\frac{Q_n(x)}{x - x_i} = \sum_{j=0}^{n-1} \bar{c}_{n,j} x^j = \frac{Q_n(x)}{x} \sum_{k=0}^{\infty} \left(\frac{x_i}{x}\right)^k = Q_n(x) \sum_{k=1}^{\infty} x_i^{k-1} x^{-k}$$

or

$$\sum_{j=0}^{n-1} \bar{c}_{n,j} x^j = \left( \sum_{\ell=0}^n c_{n,\ell} x^{\ell} \right) \left( \sum_{k=1}^{\infty} x_i^{k-1} x^{-k} \right).$$

On the right-hand side of this equation, the coefficient of  $x^j$  is given by

$$\bar{c}_{n,j} = \sum_{k=1}^{\infty} x_i^{k-1} c_{n,j+k}.$$

However, since  $c_{n,j+k} = 0$  for  $j+k > n$ ,

$$\bar{c}_{n,j} = \sum_{k=1}^{n-j} x_i^{k-1} c_{n,j+k}, \quad 0 \leq j \leq n-1.$$

The deflated polynomial can also be expressed in terms of powers of  $x/x_i$ ,

$$\frac{Q_n(x)}{x - x_i} = - \frac{Q_n(x)}{x_i} \left( \frac{1}{1 - \frac{x}{x_i}} \right) = - \frac{Q_n(x)}{x_i} \sum_{k=0}^{\infty} \left(\frac{x}{x_i}\right)^k$$

or

$$\sum_{j=0}^{n-1} \bar{c}_{n,j} x^j = - \left( \sum_{\ell=0}^n c_{n,\ell} x^{\ell} \right) \left[ \sum_{k=0}^{\infty} x_i^{-(k+1)} x^k \right].$$

This yields

$$\bar{C}_{n,j} = - \sum_{k=0}^j x_i^{-(k+1)} C_{n,j-k}, \quad 0 \leq j \leq n-1.$$

Since  $x_i$  is a root of  $Q_n(x)$ , from which it follows that  $(x - x_i)$  is a factor of  $Q_n(x)$ , all coefficients  $\bar{C}_{n,j}$  for  $j < 0$  or  $j > n-1$  must vanish. Hence, any question of convergence disappears. Either of the two formulas developed for  $\bar{C}_{n,j}$  can be used. The first formula,

$$\bar{C}_{n,j} = \sum_{k=1}^{n-j} x_i^{k-1} C_{n,j+k},$$

involves  $n-j$  terms, and the second formula,

$$\bar{C}_{n,j} = - \sum_{k=0}^j x_i^{-(k+1)} C_{n,j-k},$$

involves  $j+1$  terms. Thus, the preference of one formula over another depends on  $n$  and  $j$ . The second formula involves fewer terms when  $j < (n-1)/2$ .

Once the  $\bar{C}_{n,j}$  have been obtained, the rest of the calculations are fairly straightforward. The integral in question can be reduced to a sum:

$$\int_a^b \frac{Q_n(x)}{x - x_i} w(x) dx = \sum_{j=0}^{n-1} \bar{C}_{n,j} \int_a^b x^j w(x) dx = \sum_{j=0}^{n-1} \bar{C}_{n,j} E_j.$$

Finally, the weights are given by

$$H_i = \frac{\sum_{j=0}^{n-1} \bar{C}_{n,j} E_j}{\sum_{j=0}^{n-1} (j+1) C_{n,j+1} x_i^j}.$$

A routine to calculate  $\{x_i\}$  and  $\{H_i\}$  has been written in FORTRAN H Extended (IBM). A listing of the routine (called QINTEG) is included in appendix A. Instructions for its use are included as comment lines prefacing the body of the main subroutine.

A word about the numerical accuracy of the algorithms developed in this section is in order. QINTEG was executed to produce quadrature coefficients for several weighting functions, including those to be presented in the next section. Errors affected the accuracy of the quadrature coefficients for  $n > 10$ . The errors were not measured, but relative errors on the order of  $10^{-6}$  to  $10^{-7}$  in the abscissae and the weights were noted for certain weighting functions. (These were observable because a weighting function symmetric about the midpoint of the interval should produce abscissae and weights that also are symmetric about the midpoints.) The most likely sources of error were in the recursive computation to produce  $Q_n(x)$  and in determining the roots of  $Q_n(x)$ . The recursion is a forward formula, so errors tend to accumulate. The accuracy of the roots is suspect because, whereas the rest of the computation was done in quadruple precision (128 bits) on an IBM 370/168 computer, only double precision routines for finding roots were available. Although the question of numerical errors deserves consideration before QINTEG is extensively used, our application (sect. 4) did not place stringent demands on the algorithm. Thus, the program is presented as we used it.

This section has detailed the calculations necessary to obtain abscissae and weights for general Gaussian integration. A FORTRAN program for carrying out these calculations is presented in appendix A. Aside from the restriction from the theory that the weighting function  $w(x)$  be everywhere the same sign, a practical restriction on the weighting function is introduced in this section. The restriction is that the values of the integrals  $\int_a^b x^i w(x) dx$  for  $0 \leq i \leq 2n - 1$  must be available (or at least much easier to calculate than the original integral). To exemplify the use of QINTEG, section 4 presents the algebra necessary to remove an oscillatory factor from an integrand.

#### 4. FOURIER-LIKE INTEGRALS

This section considers integrals of the form

$$\int_a^b e^{if(x)} g(x) dx, \quad (i = \sqrt{-1})$$

where  $f(x)$  and  $g(x)$  are real-valued functions of a real variable. The real and imaginary parts are integrated separately. With the more popular integration schemes, numerical problems of efficiency arise when  $|f(b) - f(a)| \gg 2\pi$  since a large number of samples of the integrand must be taken because of the highly oscillatory nature of the exponential.

Assuming that  $f(x)$  is uniquely invertible on  $[a,b]$ , we may make the change of variables  $y = f(x)$ ,

$$\int_A^B e^{iy} g[f^{-1}(y)] [f^{-1}(y)]' dy ,$$

where  $A = f(a)$  and  $B = f(b)$ . When the integrand, apart from the exponential, is relatively slowly varying on  $(A,B)$ , absorbing the term  $e^{iy}$  in a weighting function considerably increases the efficiency for estimating the integral.

To apply the algorithms of the last section, the integral must be broken into several problems. First, let

$$M = \left\lfloor \frac{B - A}{2\pi} \right\rfloor ,$$

the number of complete cycles of the exponential in  $[A,B]$ . Then the integral is

$$\int_A^B e^{iy} F(y) dy = \int_A^{A+2\pi M} e^{iy} F(y) dy + \int_{A+2\pi M}^B e^{iy} F(y) dy ,$$

where

$$F(y) = g[f^{-1}(y)] [f^{-1}(y)]'$$

for brevity. The integral from  $A + 2\pi M$  to  $B$  is over less than one cycle of the exponential, so the real and imaginary parts can be integrated efficiently by available routines. For future reference, this integral is defined as the quantity  $I_1$ :

$$I_1 = \int_{A+2\pi M}^B e^{iy} F(y) dy .$$

For the other integral,

$$\begin{aligned}\int_A^{A+2\pi M} e^{iy} F(y) dy &= \int_0^{2\pi M} e^{i(x+A)} F(x+A) dx \\ &= e^{iA} \int_0^{2\pi M} e^{ix} F(x+A) dx\end{aligned}$$

To integrate the real or imaginary parts of this integral using  $\cos x$  or  $\sin x$  as a weighting function, one cannot apply the results of section 3, since the weighting function must be everywhere positive. However, this next integral can be evaluated

$$\int_A^{A+2\pi M} e^{iy} F(y) dy = e^{iA} \left[ \int_0^{2\pi M} (1 + i + e^{ix}) F(x+A) dx - (1+i) \int_0^{2\pi M} F(x+A) dx \right].$$

We now define

$$\begin{aligned}I_2 &= \operatorname{Re} \left\{ \int_0^{2\pi M} (1 + i + e^{ix}) F(x+A) dx \right\} \\ &= \int_0^{2\pi M} (1 + \cos x) F(x+A) dx, \\ I_3 &= \operatorname{Im} \left\{ \int_0^{2\pi M} (1 + i + e^{ix}) F(x+A) dx \right\} \\ &= \int_0^{2\pi M} (1 + \sin x) F(x+A) dx, \\ I_4 &= \int_0^{2\pi M} F(x+A) dx.\end{aligned}$$

Then the original integral is given by

$$\int_a^b e^{if(x)} g(x) dx = I_1 + e^{iA} [I_2 - I_4 + i(I_3 - I_4)].$$

The integral  $I_4$  does not involve an oscillating factor in the integrand, so it can be evaluated by standard methods, just as  $I_1$ .

The parts of the original integral still to be considered are

$$\int_0^{2\pi M} (1 + \cos x) F(x + A) dx$$

and

$$\int_0^{2\pi M} (1 + \sin x) F(x + A) dx.$$

To obtain quadrature coefficients by using  $1 + \cos x$  or  $1 + \sin x$  as weighting functions, values for  $\int_0^{2\pi M} x^j (1 + \cos x) dx$  and  $\int_0^{2\pi M} x^j (1 + \sin x) dx$  must be supplied for all integer values of  $j$  up to  $2n - 1$ , where  $n$  is the order of polynomial approximation to be used for  $F$  in  $(A, A + 2\pi M)$ . To obtain a recursion relationship for these values, one can integrate by parts for each weighting function.

For the weighting function  $1 + \cos x$ , integration by parts twice yields

$$\int x^j \cos x dx = x^j \sin x + jx^{j-1} \cos x - j(j+1) \int x^{j-2} \cos x dx,$$

valid for  $j > 1$ . Thus,

$$\begin{aligned} E_j &= \int_0^{2\pi M} x^j (1 + \cos x) dx \\ &= \int_0^{2\pi M} x^j dx + \int_0^{2\pi M} x^j \cos x dx \\ &= \frac{x^{j+1}}{j+1} \Big|_0^{2\pi M} + (x^j \sin x + jx^{j-1} \cos x) \Big|_0^{2\pi M} - j(j+1) \int_0^{2\pi M} x^{j-2} \cos x dx \\ &= \frac{(2\pi M)^{j+1}}{j+1} + j(2\pi M)^{j-1} - j(j+1) \int_0^{2\pi M} x^{j-2} \cos x dx \\ &= \frac{(2\pi M)^{j+1}}{j+1} + j(2\pi M)^{j-1} - j(j+1) \left( \int_0^{2\pi M} x^{j-2} dx + \int_0^{2\pi M} x^{j-2} \cos x dx - \int_0^{2\pi M} x^{j-2} dx \right) \\ &= \frac{(2\pi M)^{j+1}}{j+1} + j(2\pi M)^{j-1} - j(j+1) \left[ E_{j-2} - \frac{(2\pi M)^{j-1}}{j-1} \right] \end{aligned}$$

or, finally (for a weighting function  $w(x) = 1 + \cos x$ ),

$$E_j = \frac{(2\pi M)^{j+1}}{j+1} + 2j(2\pi M)^{j-1} - j(j-1)E_{j-2}, \quad j > 1.$$

For  $j = 0$  and  $j = 1$ ,  $E_j$  can be evaluated directly:

$$E_0 = \int_0^{2\pi M} (1 + \cos x) dx = 2\pi M,$$

$$E_1 = \int_0^{2\pi M} x(1 + \cos x) dx = \frac{(2\pi M)^2}{2}.$$

These relationships can be used directly to obtain quadrature coefficients for  $I_2$  using QINTEG.

The same procedure can be carried out for  $I_3$ , with a weighting function  $w(x) = 1 + \sin x$ . Integration by parts yields

$$\int x^j \sin x dx = -x^j \cos x + jx^{j-1} \sin x - j(j-1) \int x^{j-2} \sin x dx, \quad j > 1.$$

Thus,

$$\begin{aligned} E_j &= \int_0^{2\pi M} x^j (1 + \sin x) dx \\ &= \int_0^{2\pi M} x^j dx + \int_0^{2\pi M} x^j \sin x dx \\ &= \frac{x^{j+1}}{j+1} \Big|_0^{2\pi M} + \left( -x^j \cos x + jx^{j-1} \sin x \right) \Big|_0^{2\pi M} - j(j-1) \int_0^{2\pi M} x^{j-2} \sin x dx \\ &= \frac{(2\pi M)^{j+1}}{j+1} - (2\pi M)^j - j(j-1) \left( \int_0^{2\pi M} x^{j-2} dx + \int_0^{2\pi M} x^{j-2} \sin x dx - \int_0^{2\pi M} x^{j-2} dx \right) \\ &= \frac{(2\pi M)^{j+1}}{j+1} - (2\pi M)^j - j(j-1) \left[ E_{j-2} - \frac{(2\pi M)^{j-1}}{j-1} \right] \end{aligned}$$

or, finally,

$$E_j = \frac{(2\pi M)^{j+1}}{j+1} - (2\pi M)^j + j(2\pi M)^{j-1} - j(j-1)E_{j-2}, \quad j > 1.$$

For  $j = 0$  and  $j = 1$ ,

$$E_0 = \int_0^{2\pi M} (1 + \sin x) dx = 2\pi M ,$$

$$E_1 = \int_0^{2\pi M} x(1 + \sin x) dx = \frac{(2\pi M)^2}{2} - 2\pi M .$$

These relationships can be used to obtain quadrature coefficients for  $I_3$  using QINTEG.

Tables I to VI show the abscissae and the weights for 10-point Gaussian integration using these weighting functions for  $M = 10, 20, 40, 80, 160$ , and  $320$ . These coefficients were used in a numerical integration routine to evaluate Fourier-like integrals. The routine was a modification of FOGIE<sup>1</sup> and used a similar algorithm, with interval halving<sup>10</sup> and extrapolation of the sequence of approximations<sup>11</sup> to accelerate convergence. The average execution for this routine on a set of integrals with  $M$  varying from 0 to 40 was 70 times faster than FOGIE. (The integrand was of the form  $[x\{(B^2 - x^2)(x^2 - A^2)\}^{1/2}]^{-1}$ .) Thus, despite the added complexity of having to solve four integrals ( $I_1$  to  $I_4$ ), rather than two (the real and imaginary parts of the original integral), the approach used in this report produced a routine vastly more efficient than any available numerical integration routine known to the author.

<sup>1</sup>A. Hausner and J. D. Hutchinson, FOGIE: An Adaptive Code for Numerical Integrals Using Gaussian Quadrature, Proc. 1975 Army Numerical Analysis and Computer Conference, Army Research Office Report 75-3 (1975), 139-177.

<sup>10</sup>C. deBoor, On Writing an Automatic Integration Algorithm, Mathematical Software, John R. Rice, ed., Academic Press, New York (1971), 201-209.

<sup>11</sup>C. deBoor, CADRE: An Algorithm for Numerical Quadrature, Mathematical Software, John R. Rice, ed., Academic Press, New York (1971), 417-449.

TABLE I. ABSCISSAE AND WEIGHTS FOR GAUSSIAN INTEGRATION WITH WEIGHTING FUNCTION  $w(x)$ : 10 CYCLES

$w(x) = 1 + \cos x$		$w(x) = 1 + \sin x$	
Abscissa	Weight	Abscissa	Weight
0.6087374711	2.6807763508	0.8752104673	3.3695813000
4.9914285101	4.6772747988	3.7865818792	3.7269621119
10.3820693807	6.4545355569	9.4717013503	7.0544987207
17.8773971242	8.3559383986	17.3092678040	8.5022382935
26.7555983405	9.2474011736	26.2502048248	9.2561454163
36.0762547313	9.2474011736	35.5563915932	9.2184344844
44.9544559472	8.3559384005	44.4130198287	8.3639907442
52.4497836920	6.4545355550	52.0734054498	6.9040953344
57.8404245609	4.6772747996	58.3063199301	5.6632258042
62.2231156010	2.6807766076	62.2510452612	0.7726807607

TABLE II. ABSCISSAE AND WEIGHTS FOR GAUSSIAN INTEGRATION WITH WEIGHTING FUNCTION  $w(x)$ : 20 CYCLES

$w(x) = 1 + \cos x$		$w(x) = 1 + \sin x$	
Abscissa	Weight	Abscissa	Weight
1.0958562160	3.8509782861	1.4073606532	5.3746551037
8.2760859072	9.5999529851	8.6742392405	9.3783293959
20.0439789573	13.8180757985	20.2902602865	13.7154348092
35.5441895125	16.9568219659	35.7047378274	16.8806675057
53.4589364414	18.6060242108	53.5465988868	18.5354461291
72.2047697025	18.6060242108	72.2239429478	18.5399805836
90.1195166298	16.9568219727	90.0760264722	16.8978671035
105.6197271888	13.8180757925	105.5191712220	13.7582591895
117.3876202339	9.5999529878	117.1642946953	9.2992491276
124.5678499286	3.8509781097	123.3816739078	3.2838176515

TABLE III. ABSCISSAE AND WEIGHTS FOR GAUSSIAN INTEGRATION WITH  
WEIGHTING FUNCTION  $w(x)$ : 40 CYCLES

$w(x) = 1 + \cos x$		$w(x) = 1 + \sin x$	
Abcissa	Weight	Abcissa	Weight
2.8183314516	7.8585900665	2.6565132959	8.9339537867
16.4855928817	18.8804230134	16.5290556985	18.7939382658
39.9253416820	27.6523406643	39.8338445222	27.4824982790
70.9730313642	33.9801300369	70.6882254019	33.7669783574
106.8772903925	37.2922221234	106.3662919383	37.0563538843
144.4501218949	37.2922221234	143.7002563722	37.0535973607
180.3543809222	33.9801300425	179.3720458652	33.7562834074
211.4020706062	27.6523406611	210.2072642327	27.4506937547
234.8418194051	18.8804230133	233.4447939120	18.6623839329
248.5090808355	7.8585903048	246.9679666228	8.3707313204

TABLE IV. ABSCISSAE AND WEIGHTS FOR GAUSSIAN INTEGRATION WITH  
WEIGHTING FUNCTION  $w(x)$ : 80 CYCLES

$w(x) = 1 + \cos x$		$w(x) = 1 + \sin x$	
Abcissa	Weight	Abcissa	Weight
6.2942643908	16.4238170553	5.7276955655	16.9957022744
33.5952185179	37.6003753926	33.1561465055	37.5586612404
80.3173832187	55.1422003060	79.7934759820	55.0262214933
142.2390857322	67.7755507792	141.5795832699	67.6238618053
213.8546984578	74.3854676502	213.0334404429	74.2164516187
288.8001261154	74.3854676502	287.8080856863	74.2156211234
360.4157388472	67.7755507526	359.2600654194	67.6206860473
422.3374413443	55.1422003333	421.0406014177	55.0171626627
469.0596060704	37.6003753744	467.6598685263	37.5250478206
496.3605601771	16.4238181702	495.0025963240	16.8554086052

TABLE V. ABSCISSAE AND WEIGHTS FOR GAUSSIAN INTEGRATION WITH WEIGHTING  
FUNCTION  $w(x)$ : 160 CYCLES

$w(x) = 1 + \cos x$		$w(x) = 1 + \sin x$	
Abcissa	Weight	Abcissa	Weight
12.9796420057	33.3360123661	12.1936860013	33.6189092406
67.6552243405	75.1396513709	66.9327500610	75.1180427176
161.0067973988	110.1674325314	160.2388236537	110.1045290787
284.7166120591	135.4035687063	283.8747879464	135.3207702919
427.7914246520	148.6081584499	426.8612580903	148.5158506231
577.5182244980	148.6081584499	576.4948799015	148.5156416467
720.5930370840	135.4035687338	719.4808781000	135.3199727513
844.3028517811	110.1674325082	843.1154468964	110.1022657342
937.6544247963	75.1396513680	936.4170389156	75.1097518556
992.3300071481	33.3360135411	991.1349134249	33.5839141270

TABLE VI. ABSCISSAE AND WEIGHTS FOR GAUSSIAN INTEGRATION WITH  
WEIGHTING FUNCTION  $w(x)$ : 320 CYCLES

$w(x) = 1 + \cos x$		$w(x) = 1 + \sin x$	
Abcissa	Weight	Abcissa	Weight
26.1632634886	66.9356733119	25.2687701457	67.0746046124
135.5658268848	150.2531469489	134.7018077959	150.2417920174
322.2213469950	220.2712914684	321.3339328777	220.2390735954
569.5670711729	270.7247214675	568.6418609881	270.6823636132
855.6289767982	297.1248171845	854.6585865153	297.0776244037
1154.9903215015	297.1248171845	1153.9723172997	297.0775722417
1441.0522271171	270.7247215075	1439.9889249782	270.6821644543
1688.3979513144	220.2712914419	1687.2965048290	220.2385090728
1875.0534714023	150.2531469488	1873.9275134964	150.2397288068
1984.4560348127	66.9356720769	1983.3552710669	67.0658633200

## 5. SUMMARY

This report presents a program to compute Gaussian quadrature coefficients for integration with respect to a weighting function. The purpose of developing this program was to combat the inefficiency of existing numerical integration routines in evaluating integrals with oscillatory integrands. However, the technique of absorbing undesirable behavior of integrands into the weighting function can be applied to many other problems. Specifically, discontinuities or singularities within the interval of integration generally preclude the use of many programs and produce unreliable results in most other programs. When such behavior can be absorbed in the weighting function, the coefficients computed by QINTEG can be used to produce reliable estimates more efficiently.

The program was written to be usable in a variety of problems. The user must allocate storage in the calling program and supply one subroutine to QINTEG. That subroutine must compute  $E_j$ --the moments of the weighting function. Then QINTEG returns to the calling program the abscissae and the weights for estimating integrals. The only restrictions on the weighting function (for finite intervals) are that it be nonnegative (or equivalently, nonpositive) throughout the interval and that  $E_j$  must all exist.

This technique was applied to the integration of Fourier-like integrals. When compared with the time of an existing numerical integration routine, FOGIE, the execution time decreased by a factor of 70. Since FOGIE is fairly efficient,<sup>1</sup> in many problems the benefits of using QINTEG more than balance the difficulties of understanding its use.

## ACKNOWLEDGEMENTS

I thank Art Hausner, the developer of FOGIE, for many long and helpful hours of discussion. Without the use of his ear (a colleague's ear is, I believe, a programmer's principal debugging aid), the development of QINTEG would have been a much more arduous task.

I thank also Dennis McGuire. To him is due the original problem of calculating coherence properties of reflected laser light. Many more hours of discussion were spent with Dennis while I tried to understand the behavior of integrands that arose in the calculations. These discussions convinced me that existing integration routines were impractical and that an improvement such as QINTEG was necessary.

---

<sup>1</sup>A. Hausner and J. D. Hutchinson, FOGIE: An Adaptive Code for Numerical Integrals Using Gaussian Quadrature, Proc. 1975 Army Numerical Analysis and Computer Conference, Army Research Office Report 75-3 (1975), 139-177.

# LITERATURE CITED

- (1) A. Hausner and J. D. Hutchinson, FOGIE: An Adaptive Code for Numerical Integrals Using Gaussian Quadrature, Proc. 1975 Army Numerical Analysis and Computer Conference, Army Research Office Report 75-3 (1975), 139-177.
- (2) Petr Beckmann, Orthogonal Polynomials for Engineers and Scientists, The Golem Press, Boulder, CO (1973).
- (3) M. Abramowitz and I. A. Stegun ed., Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, National Bureau of Standards Applied Mathematics Series, 55 (1970).
- (4) P. J. Davis and P. Rabinowitz, Numerical Integration, Blaisdell, Waltham, MA (1967).
- (5) J. W. Tukey, The Estimation of (Power) Spectra and Related Quantities, in On Numerical Approximation, R. E. Langer, ed., University of Wisconsin Press, Madison, WI (1959), 389-411.
- (6) W. Gautschi, Algorithm 331--Gaussian Quadrature Formulas, CACM, 11, No. 6 (June 1968).
- (7) W. Gautschi, Construction of Gauss-Christoffel Quadrature Formulas, Mathematics of Computation, 22 (1968), 251-270.
- (8) IMSL Library 1, Editor 6 (FORTRAN IV), IBM S/370-360, Xerox Sigma S/6-7-9-11-560 (1977).
- (9) IBM System/360 Scientific Subroutine Package, Version III, Programmers Manual, IBM GH20-0205-4 (August 1970).
- (10) C. deBoor, On Writing an Automatic Integration Algorithm, Mathematical Software, John R. Rice, ed., Academic Press, New York (1971), 201-209.
- (11) C. deBoor, CADRE: An Algorithm for Numerical Quadrature, Mathematical Software, John R. Rice, ed., Academic Press, New York (1971), 417-449.

## APPENDIX A.--QINTEG COMPUTER PROGRAM LISTING

This appendix lists computer program QINTEG that computes Gaussian quadrature coefficients for integration with respect to a weighting function.

# APPENDIX A

C	SUBROUTINE QINTEG(N,X,W,C,E,H,C1,R,USERE)	00000100
C		00000200
C	PURPOSE	00000300
C	THIS SUBROUTINE COMPUTES ABSCISSAE AND WEIGHTS FOR GAUSSIAN	00000400
C	INTEGRATION OVER AN ARBITRARY INTERVAL WITH A USER-DETERMINED	00000500
C	WEIGHTING FUNCTION. THAT IS, IT COMPUTES X(I) AND W(I),	00000600
C	I=1,...,N TO APPROXIMATE	00000700
C		00000800
C	B	00000900
C	/                  N	00001000
C	I	00001100
C	I F(X)G(X) DX = SUM W(I)G(X(I))	00001200
C	/	00001300
C	I=1	00001400
C	A	00001500
C		00001600
C	USAGE	00001700
C	THIS SUBROUTINE IS INVOKED WITH THE SEQUENCE	00001800
C	REAL*16 X,W,C,E,H,R	00001900
C	REAL*8 C1	00002000
C	EXTERNAL USERE	00002100
C	.	00002200
C	.	00002300
C	.	00002400
C	CALL QINTEG(N,X,W,C,E,H,C1,R,USERE)	00002500
C		00002600
C	ARGUMENTS	00002700
C	N - INTEGER*4 INPUT SCALAR. THE DESIRED NUMBER OF	00002800
C	SUMMATION POINTS AND WEIGHTS.	00002900
C	X - REAL*16 OUTPUT ARRAY. THE SET OF ABSCISSAE.	00003000
C	MUST BE DIMENSIONED N IN THE CALLING PROGRAM.	00003100
C	W - REAL*16 OUTPUT ARRAY. THE SET OF WEIGHTS.	00003200
C	MUST BE DIMENSIONED N IN THE CALLING PROGRAM.	00003300
C	C - REAL*16 WORKING ARRAY. MUST BE DIMENSIONED	00003400
C	(N+1)(N+2)/2 IN THE CALLING PROGRAM.	00003500
C	E - REAL*16 WORKING ARRAY. MUST BE DIMENSIONED	00003600
C	2N IN THE CALLING PROGRAM.	00003700
C	H,R - REAL*16 WORKING ARRAYS. MUST BE DIMENSIONED	00003800
C	N IN THE CALLING PROGRAM.	00003900
C	C1 - REAL*8 WORKING ARWAY. MUST BE DIMENSIONED (N+1)	00004000
C	IN THE CALLING PROGRAM.	00004100
C	USERE - USER SUPPLIED ROUTINE NAME. USED TO COMPUTE	00004200
C	VALUES OF ARRAY 'E' AS DESCRIBED BELOW. THE	00004300
C	NAME OF 'USERE' MUST BE DECLARED EXTERNAL IN	00004400
C	THE CALLING PROGRAM.	00004500
C		00004600
C	SUBROUTINES USED (* INDICATES NOT INCLUDED IN THIS MODULE)	00004700
C	POLY	00004800
C	*ZPOLR (IMSL DOUBLE PRECISION LIBRARY)	00004900
C	*'USERE'	00005000
C		00005100
C	SUBROUTINE 'USERE' IS DEFINED BY THE USER.	00005200
C	IT IS CALLED BY 'POLY' WITH THE CALLING	00005300
C	SEQUENCE:	00005400
C	CALL USERE(M,E,ICODE) WHERE M IS SET TO 2*N.	00005500
C	'USERE' MUST HAVE THE FOLLOWING PROGRAMMING:	00005600
C	SUBROUTINE USERE(M,E,ICODE)	00005700
C	REAL*16 E(M)	00005800
C		00005900

## APPENDIX A

```

C                                     00006000
C                                     00006100
C      RETURN                        00006200
C      END                          00006300
C      'USERE' MUST COMPUTE VALUES OF ARRAY 'E' SUCH THAT: 00006400
C      IF ICODE=0 ON RETURN, ELEMENT E(I) CONTAINS THE INTEGRAL 00006500
C      OF  $F(X)*(X**(I-1))$  OVER THE INTERVAL OF INTEGRATION FOR 00006600
C       $1 \leq I \leq M$ . 00006700
C      IF ICODE>0 ON RETURN, ELEMENTS E(1),...,E(M/2) CONTAIN 00006800
C      THE INTEGRATION ABSCISSAE AND ELEMENTS E(M/2+1),...,E(M) 00006900
C      CONTAIN THE INTEGRATION WEIGHTS. 00007000
C      IF 'USERE' RETURNS ICODE=0, THE REST OF THIS SUBROUTINE 00007100
C      COMPUTES THE WEIGHTS AND ABSCISSAE. 00007200
C                                     00007300
C      PROGRAM NOTES 00007400
C      THE PURPOSE OF THE RETURN CODE FOR 'USERE' IS TO ALLOW THE 00007500
C      WORK DONE IN THIS SUBROUTINE TO BE DONE NO MORE OFTEN THAN 00007600
C      NECESSARY. ONCE THIS SUBROUTINE COMPUTES THE WEIGHTS AND 00007700
C      ABSCISSAE FOR A PARTICULAR COMBINATION OF PARAMETERS (PASSED 00007800
C      TO 'USERE' IN COMMON FROM THE MAIN PROGRAM), THIS INFORMATION 00007900
C      CAN, IN CERTAIN APPLICATIONS, ALSO BE PASSED TO 'USERE' SO THE 00008000
C      OVERHEAD OF PERFORMING THE CALCULATIONS CAN BE AVOIDED WHEN 00008100
C      THE RESULTS ARE NEEDED AGAIN. 00008200
C                                     00008300
C      RESTRICTIONS 00008400
C      THE WEIGHTING FUNCTION F(X) MAY NOT CHANGE SIGN OVER 00008500
C      THE INTERVAL OF INTEGRATION. 00008600
C                                     00008700
C      REFERENCES 00008800
C      BECKMAN, P., "ORTHOGONAL FUNCTIONS FOR ENGINEERS AND 00008900
C      PHYSICISTS," GOLEM PRESS, BOULDER, COL., 1973. 00009000
C      HDL LABORATORY NOTEBOOK 7609, PP. 34-42. 00009100
C                                     00009200
C      AUTHOR AND MODIFICATIONS 00009300
C      TED HOPP 7/24/78 00009400
C      8/08/78 - INCLUDE OPTION FOR TABLE LOOKUP 00009500
C      OF ABSCISSAE AND WEIGHTS. 00009600
C                                     00009700
C      REAL*16 C(1),E(1),H(1),X(1),W(1),Q 00009800
C      REAL*8 C1(1) 00009900
C      COMPLEX*16 R(1) 00010000
C                                     00010100
C      CALCULATE COEFFICIENTS AND E ARRAY FOR N-TH DEGREE POLYNOMIAL 00010200
C      CALL USERE(2*N,E,ICODE) 00010300
C      IF (ICODE.GT.0) GO TO 60 00010400
C      CALL POLY(N,C,E,H,W) 00010500
C                                     00010600
C      CALCULATE ROOTS OF POLYNOMIAL - COPY COEFFICIENTS INTO DOUBLE 00010700
C      PRECISION ARRAY C1 00010800
C      N1=N+1 00010900
C      DO 5 I=1,N1 00011000
C      C1(I)=C(I) 00011100
C      CONTINUE 00011200
C      CALL ZPOLR(C1,N,R,IER) 00011300
C      IF (IER.LT.129) GO TO 10 00011400
C      WRITE (6,6000) IER,(R(I),I=1,N) 00011500
C      6000 FORMAT (' IER=',IS,'. ROOTS:',(1X,1P2D15.8)) 00011600
C      00011700
C      00011800

```

# APPENDIX A

C	STOP	00011900
C	FOR EACH ROOT DO THE FOLLOWING	00012000
C		00012100
10	DO 50 I=1,N	00012200
	X(I)=DREAL(R(I))	00012300
C		00012400
C	DEFLATE POLYNOMIAL - USE MOST EFFICIENT CALCULATION FOR EACH	00012500
C	COEFFICIENT.	00012600
C		00012700
	J2=N/2	00012800
	DO 20 J=1,J2	00012900
	H(J)=0.00	00013000
	DO 20 K=1,J	00013100
	L=N+2-K	00013200
	H(J)=(H(J)-C(L))/X(I)	00013300
20	CONTINUE	00013400
C		00013500
C	CHANGE FORMULA FOR COEFFICIENTS	00013600
C		00013700
	J2=J2+1	00013800
	J3=N-1	00013900
	DO 30 J=J2,J3	00014000
	H(J)=C(1)	00014100
	K1=N-J+1	00014200
	DO 30 K=2,K1	00014300
	H(J)=X(I)*H(J)+C(K)	00014400
30	CONTINUE	00014500
	H(N)=C(1)	00014600
C		00014700
C	EVALUATE WEIGHT	00014800
C		00014900
	Q=C(1)*QFLOAT(N)	00015000
	W(1)=H(1)*E(1)	00015100
	DO 40 J=2,N	00015200
	Q=X(1)*Q+C(J)*QFLOAT(N-J+1)	00015300
	W(1)=W(1)+H(J)*E(J)	00015400
40	CONTINUE	00015500
	W(1)=W(1)/O	00015600
50	CONTINUE	00015700
	RETURN	00015800
C		00015900
C	ABSCISSAE AND WEIGHTS SUPPLIED BY 'USERE'	00016000
C		00016100
60	DO 70 J=1,N	00016200
	K=N+J	00016300
	X(J)=E(J)	00016400
	W(J)=E(K)	00016500
70	CONTINUE	00016600
	RETURN	00016700
	END	00016800
	SUBROUTINE POLY(N,C,E,H,B)	00016900
C		00017000
C	PURPOSE	00017100
C	THIS PROGRAM COMPUTES COEFFICIENTS OF ORTHOGONAL POLYNOMIALS.	00017200
C	SEE LABORATORY NOTEBOOK 7609 FOR DETAILS OF THE DERIVATION OF	00017300
C	THIS CODE.	00017400
C		00017500
C		00017600
C	USAGE	00017700

```

C      CALL POLY(N,C,E,H,B)                                00017800
C                                                         00017900
C      ARGUMENTS                                           00018000
C      N - INTEGER*4 INPUT SCALAR. THE DEGREE OF THE DESIRED ORTHOGONAL 00018100
C      POLYNOMIAL.                                         00018200
C      C - REAL*16 OUTPUT ARRAY. ON OUTPUT, C(1)...C(N+1) CONTAIN THE 00018300
C      COEFFICIENTS OF THE N-TH DEGREE POLYNOMIAL IN ORDER OF 00018400
C      DECREASING POWER OF THE VARIABLE. THIS ARRAY MUST BE 00018500
C      DIMENSIONED (N+1)(N+2)/2 IN THE CALLING PROGRAM.    00018600
C      E - REAL*16 INPUT ARRAY. ELEMENTS OF THIS ARRAY ARE COMPUTED 00018700
C      BEFORE THIS SUBROUTINE IS CALLED. THE ELEMENT E(I) 00018800
C      CONTAINS THE INTEGRAL OVER THE ORTHOGONALITY INTERVAL 00018900
C      OF  $W(X) \cdot (X^{**}(I-1))$ , WHERE  $W(X)$  IS THE ORTHOGONALITY 00019000
C      WEIGHTING FUNCTION. TO COMPUTE THE N-TH DEGREE POLY- 00019100
C      NOMIAL, VALUES OF E(1)...E(2N) MUST BE SUPPLIED. 00019200
C      H - REAL*16 WORKING ARRAY. THIS ARRAY IS USED AS WORKING 00019300
C      STORAGE. IT MUST HAVE AT LEAST N ELEMENTS. (ON OUTPUT, 00019400
C      THE VALUE OF H(I) WILL BE THE NORM OF THE (I-1)-ST 00019500
C      DEGREE ORTHOGONAL POLYNOMIAL.) 00019600
C      B - REAL*16 WORKING ARRAY. THIS ARWAY IS USED AS WORKING 00019700
C      STORAGE. IT MUST HAVE AT LEAST N ELEMENTS. 00019800
C                                                         00019900
C      SUBROUTINES USED (* INDICATES NOT INCLUDED IN THIS MODULE) 00020000
C      NONE 00020100
C                                                         00020200
C      PROGRAM NOTES 00020300
C      STORAGE IN ARRAYS C, H, AND B IS AS FOLLOWS: 00020400
C      C - THE COEFFICIENT OF  $X^{**}J$  IN THE I-TH ORTHOGONAL POLYNOMIAL 00020500
C      IS STORED IN C(I+J+1*(I+1)/2) DURING EXECUTION OF THIS 00020600
C      ROUTINE. BEFORE EXIT, THE COEFFICIENTS OF THE N-TH 00020700
C      DEGREE POLYNOMIAL ARE MOVED TO C(1),...,C(N+1) IN ORDER 00020800
C      OF DECREASING POWER OF X. 00020900
C      H - THE VALUE OF H FOR THE I-TH ORTHOGONAL POLYNOMIAL IS 00021000
C      STORED IN H(I+1). 00021100
C      B - THE VALUE OF B FOR THE I-TH ORTHOGONAL POLYNOMIAL IS 00021200
C      STORED IN B(I+1). 00021300
C                                                         00021400
C      REAL*16 H(1),B(1),E(1),C(1) 00021500
C                                                         00021600
C      INITIALIZE FOR CALCULATING COEFFICIENTS 00021700
C                                                         00021800
C      C(1)=1.00 00021900
C      H(1)=E(1) 00022000
C      B(1)=E(2)/H(1) 00022100
C      C(2)=-B(1) 00022200
C      C(3)=1.00 00022300
C      I=1 00022400
C                                                         00022500
C      BEGINNING OF RECURSION LOOP - POLYNOMIAL OF DEGREE I OBTAINED 00022600
C                                                         00022700
C      10 I1=I+1 00022800
C      IF (I.GE.N) GO TO 50 00022900
C                                                         00023000
C      COMPUTE VALUE OF H FOR POLYNOMIAL OF DEGREE I 00023100
C                                                         00023200
C      H(I1)=0.00 00023300
C      DO 20 J2=1,I1 00023400
C      L=I+J2 00023500
C      M=J2+(I*I1)/2 00023600

```

# APPENDIX A

	H(I1)=H(I1)+C(M)*E(L)	00023700
20	CONTINUE	00023800
C		00023900
C	COMPUTE VALUE OF B FOR POLYNOMIAL OF DEGREE I	00024000
C		00024100
	B(I1)=0.00	00024200
	DO 30 J2=1,I1	00024300
	L=J2+I1	00024400
	M=J2+(I+I1)/2	00024500
	B(I1)=B(I1)+C(M)*E(L)	00024600
30	CONTINUE	00024700
	M=(I*(I+3))/2	00024800
	B(I1)=B(I1)/H(I1)+C(M)	00024900
C		00025000
C	COMPUTE COEFFICIENTS FOR NEXT HIGHER DEGREE POLYNOMIAL	00025100
C		00025200
	J1=1	00025300
	I=I1	00025400
	L=1+(I*(I+1))/2	00025500
	M=1+(I*(J1))/2	00025600
	J=1+(J1*(I-2))/2	00025700
	C(L)=-B(I)*C(M)-H(I)*C(J)/H(J1)	00025800
	DO 40 J2=2,J1	00025900
	L=J2+(I*(I+1))/2	00026000
	M=J2+(I*J1)/2	00026100
	J=J2+(J1*(I-2))/2	00026200
	C(L)=C(M-1)-B(I)*C(M)-H(I)*C(J)/H(I-1)	00026300
40	CONTINUE	00026400
	L=(I*(I+3))/2	00026500
	M=(I*(I+1))/2	00026600
	C(L)=C(M-1)-B(I)*C(M)	00026700
	C(L+1)=C(M)	00026800
	GO TO 10	00026900
C		00027000
C	PLACE COEFFICIENTS FOR N-TH DEGREE POLYNOMIAL IN C(1)...C(N+1)	00027100
C	IN DESCENDING ORDER	00027200
C		00027300
50	DO 60 J2=1,I1	00027400
	L=2-J2+(N*(N+3))/2	00027500
	C(J2)=C(L)	00027600
60	CONTINUE	00027700
	RETURN	00027800
	END	00027900

# DISTRIBUTION

ADMINISTRATOR  
DEFENSE DOCUMENTATION CENTER  
ATTN DDC-TCA (12 COPIES)  
CAMERON STATION, BUILDING 5  
ALEXANDRIA, VA 22314

COMMANDER  
US ARMY RSCH & STD GP (EUR)  
ATTN LTC JAMES M. KENNEDY, JR.  
CHIEF, PHYSICS & MATH BRANCH  
FPO NEW YORK 09510

COMMANDER  
US ARMY MATERIEL DEVELOPMENT &  
READINESS COMMAND  
ATTN DRXAM-TL, HQ TECH LIBRARY  
5001 EISENHOWER AVENUE  
ALEXANDRIA, VA 22333

COMMANDER  
US ARMY ARMAMENT MATERIEL  
READINESS COMMAND  
ATTN DRSAR-ASF, FUZE &  
MUNITIONS SUPPORT DIV  
ATTN DRSAR-LEP-L, TECHNICAL LIBRARY  
ROCK ISLAND, IL 61299

COMMANDER  
US ARMY MISSILE & MUNITIONS  
CENTER & SCHOOL  
ATTN ATSK-CTD-F  
REDSTONE ARSENAL, AL 35809

DIRECTOR  
US ARMY MATERIEL SYSTEMS  
ANALYSIS ACTIVITY  
ATTN DRXS-YP  
ABERDEEN PROVING GROUND, MD 21005

DIRECTOR  
US ARMY BALLISTIC RESEARCH LABORATORY  
ATTN DRDAR-TSB-S (STINFO)  
ABERDEEN PROVING GROUND, MD 21005

TELEDYNE BROWN ENGINEERING  
CUMMINGS RESEARCH PARK  
ATTN DR. MELVIN L. PRICE, MS-44  
HUNTSVILLE, AL 35807

ENGINEERING SOCIETIES LIBRARY  
345 EAST 47TH STREET  
ATTN ACQUISITIONS DEPARTMENT  
NEW YORK, NY 10017

COMMANDING OFFICER  
NAVAL TRAINING EQUIPMENT CENTER  
ATTN TECHNICAL LIBRARY  
ORLANDO, FL 32813

MR. CARL DeBOOR  
MATHEMATICS RESEARCH CENTER  
UNIVERSITY OF WISCONSIN  
MADISON, WI 53706

MR. GENE H. GOLUB  
COMPUTER SCIENCE DEPARTMENT  
STANFORD UNIVERSITY  
STANFORD, CA 94305

US ARMY ELECTRONICS RESEARCH  
& DEVELOPMENT COMMAND  
ATTN WISEMAN, ROBERT S., DR., DRDEL-CT

HARRY DIAMOND LABORATORIES  
ATTN 00100, COMMANDER/TECHNICAL DIR/TSO  
ATTN CHIEF, DIV 10000  
ATTN CHIEF, DIV 20000  
ATTN CHIEF, DIV 30000  
ATTN CHIEF, DIV 40000  
ATTN RECORD COPY, 81200  
ATTN HDL LIBRARY, 81100 (3 COPIES)  
ATTN HDL LIBRARY, 81100 (WOODBIDGE)  
ATTN TECHNICAL REPORTS BRANCH, 81300  
ATTN CHAIRMAN, EDITORIAL COMMITTEE  
ATTN SANN, K. H., 11100  
ATTN GRIFFIN, J. R., 13300  
ATTN SOKOLOSKI, M. M., 00210  
ATTN CHIEF, 13000  
ATTN SZTANKAY, G., 13300  
ATTN NEMARICH, J., 13300  
ATTN MCGUIRE, D., 13300 (5 COPIES)  
ATTN PARKER, M., 11140  
ATTN HOPP, T.